CSC 3520
Machine Learning
Florida Southern College

HW2: Decision Trees
Due: Tuesday, October 11, 2022

In this assignment, you will have the opportunity to:

- manually compute information gain on a simple dataset to learn a full decision tree
- demonstrate a solid understanding of how to depict decision trees and corresponding boundaries
- write Python code to train and test decision trees for a real-world problem

1. Suppose you want to learn a decision tree from a simple dataset comprising 8 samples, each with 3 binary attributes (X_1, X_2, X_3) and a binary class label (Y).

Instance	X_1	X_2	X_3	Y
1	0	0	0	0
2	0	0	1	0
3	0	1	0	1
4	0	1	1	1
5	1	0	1	1
6	1	0	1	1
7	1	1	0	0
8	1	1	0	0

(a) Compute the entropy of the class label.

(b) Calculate the information gain for each of the three attributes.

(c) Which attribute should be selected for the root of the decision tree? Why?

(d) After the root node is selected, the entire tree can be learned by recursively splitting the data into subgroups, finding the next best attribute to split on, dividing the subgroup into smaller groups, and so on. How do you know when to stop growing the tree (i.e. what are the stopping criteria)?

(e) By manually running the algorithm described above, draw the resulting tree.

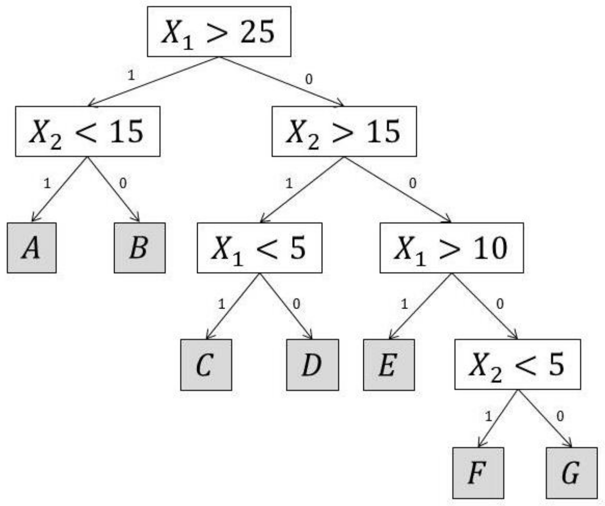
(f) Compute the training error.

(g) Now suppose you are presented new instances for which the class Y is unknown. Use your decision tree to predict the label of each instance listed below.

Instance	X_1	X_2	X_3	Y
9	1	1	1	?
10	1	0	0	?
11	0	1	1	?

(h) Do you have any basis on which to evaluate if the tree is overfitting? Why or why not? How might you combat overfitting in a decision tree?

2. Consider the following decision tree.



(a) Draw the decision boundaries defined by this tree in 2D space (X_1, X_2). Each leaf is labeled with a letter. Write this letter in the corresponding region of the instance space.

(b) Draw another decision tree that is syntactically different from the one shown above but defines the same decision boundaries.

3. Parkinson's disease (PD) is a common progressive disorder of the central nervous system that affects millions of people around the world. The exact cause of PD is unknown and there is no known cure at this time. Treatment is usually focused on mitigating symptoms that include tremors, movement loss, and speech impairment. Recently, researchers have discovered that certain measurements of dysphonia (difficulty in speaking) can help discriminate between healthy individuals and those suffering from PD. In this problem, you will use decision trees to predict whether someone has Parkinson's disease based on dysphonia measurements.

The data for this problem has been provided for you. There are 185 samples for training and 10 samples for testing. Each sample contains 22 attributes related to dysphonia; for details about these measurements, see https://archive.ics.uci.edu/ml/datasets/Parkinsons or [1]. The relevant data files are listed below.

`attributes.txt`	List of dysphonia measurement variables used for classification.
`trainingdata.txt` `testingdata.txt`	Dysphonia measurement values for each training/testing sample. Each row is a unique sample containing 22 comma-separated floating-point values.
`traininglabels.txt` `testinglabels.txt`	Each line corresponds to a sample from the training/testing set and determines whether that individual is healthy (0) or has Parkinson's (1).

[1] Little, M.A., McSharry, P.E., Hunter, E.J., Spielman, J. and Ramig, L.O., 2009. Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. *IEEE Transactions on Biomedical Engineering*, 56(4), pp.1015-1022.

Write a Python program (`decisiontree.py`) to train and test a decision tree using information gain on the data described above. This program has already been started for you.

NOTES:

- You do not need to add/remove/modify the imported modules at the top of the program.
- The code is broken down into tasks (see the comments). This is meant to help you, but you may diverge from this structure as long as your program runs and can be used to respond to the questions below.
- Google is your friend! Try searching for documentation related to the imported modules/functions. If you need a good place to start, try this:
  http://scikit-learn.org/dev/modules/generated/sklearn.tree.DecisionTreeClassifier.html
  https://scikit-learn.org/stable/modules/generated/sklearn.tree.plot_tree.html

In your homework submission, answer the following questions:

(a) What does the decision tree look like? Show a visualization of the full learned tree using `matplotlib` and `plot_tree`. In your visualization, include feature names, class names, filled nodes, and rounded nodes.

(b) What is the depth of the tree? Consider the depth to be the maximum number of decision nodes (do not count leaf nodes) that could be used to classify a single example.

(c) How many leaf nodes are there?

(d) What is the name of the attribute used to split the data at the root node of the tree?

(e) Compute the information gain for the attribute listed above by hand.
HINT: If you visualized the tree correctly, you should have all the information you need in the visualization.

(f) How many of the 22 attributes are used in the tree? What does this mean about the other attributes?

(g) Consider the confusion matrix for this tree on the test data. Does the tree more often misclassify people who are healthy or people who have Parkinson's? Practically speaking, which bias would you rather have (misclassifying healthy or sick people)? Explain your reasoning.

(h) Does your decision tree overfit your data? How do you know?